

EXHIBIT G

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

U.S. Patent No. 7,212,811 – Nordstrom, Inc.

Claims 1 and 5

Push Data LLC (“Push Data”) provides evidence of infringement of Claims 1 and 5 of U.S. Patent No. 7,212,811 (hereinafter “the ‘811 patent”) by Nordstrom, Inc. (“Nordstrom”). In support thereof, Push Data provides the following claim charts.

“Accused Instrumentalities” as used herein refers to at least the Nordstrom application as developed for mobile electronic devices. These claim charts demonstrate Nordstrom’s infringement, and provide notice of such infringement, by comparing each element of the asserted claims to corresponding components, aspects, and/or features of the Accused Instrumentalities. These claim charts are not intended to constitute an expert report on infringement. These claim charts include information provided by way of example, and not by way of limitation.

The analysis set forth below is based only upon information from publicly available resources regarding the Infringing Instrumentalities, as Nordstrom has not yet provided any non-public information. An analysis of Nordstrom’s (or other third parties’) technical documentation and/or software source code may assist in fully identify all infringing features and functionality. Accordingly, Push Data reserves the right to supplement this infringement analysis once such information is made available to Push Data. Furthermore, Push Data reserves the right to revise this infringement analysis, as appropriate, upon issuance of a court order construing any terms recited in the asserted claims.

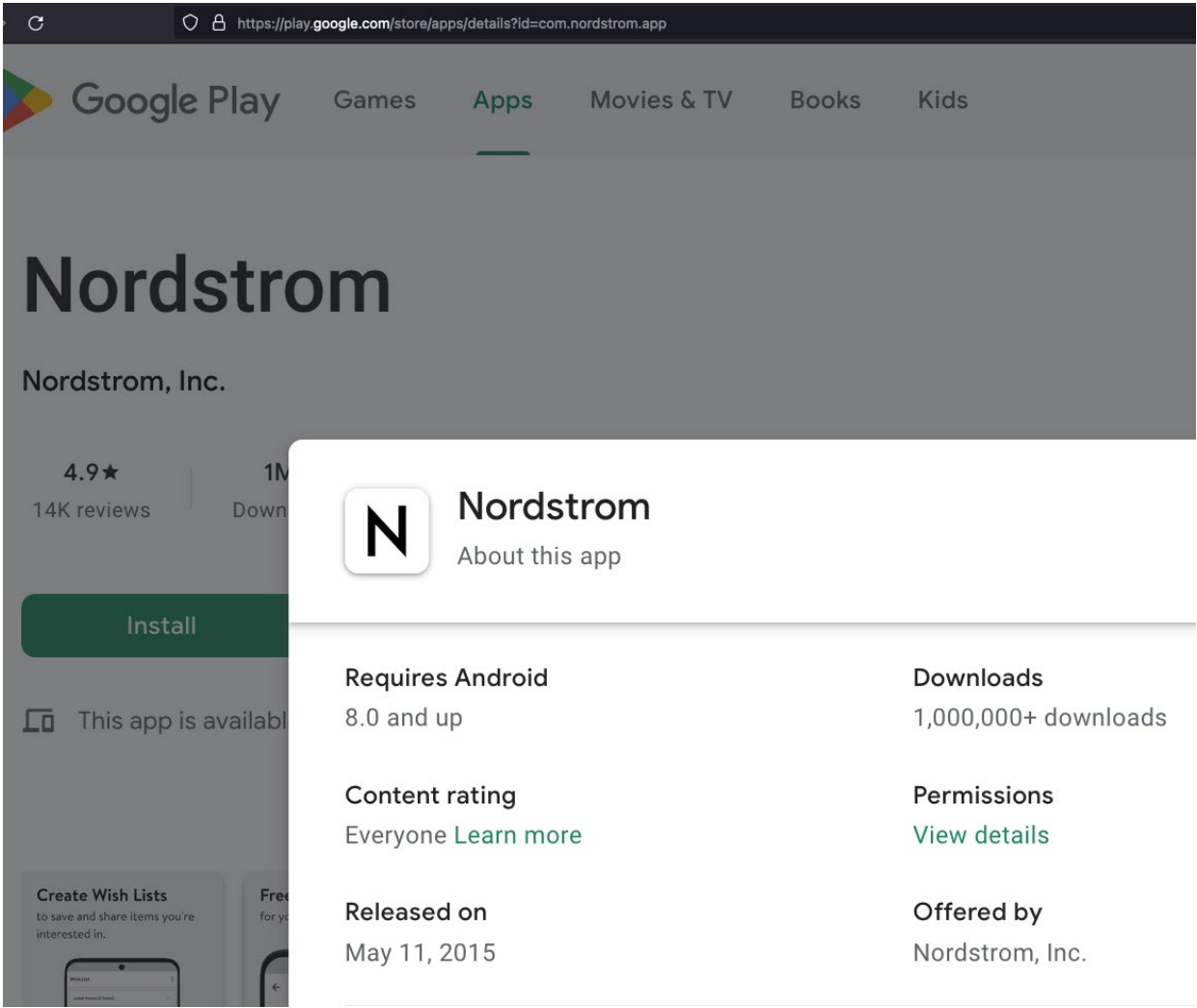
Unless otherwise noted, Push Data contends that Nordstrom directly infringes the ‘811 patent in violation of 35 U.S.C. § 271(a) by selling, offering to sell, making, using, and/or importing the Infringing Instrumentalities. The following exemplary analysis demonstrates that infringement.

Unless otherwise noted, Push Data believes and contends that each element of each claim asserted herein is literally met through Nordstrom’s provision of the Infringing Instrumentalities. However, to the extent that Nordstrom attempts to allege that any asserted claim element is not literally met, Push Data believes and contends that such elements are met under the doctrine of equivalents. More specifically, in its investigation and analysis of the Infringing Instrumentalities, Push Data did not identify any substantial differences between the elements of the patent claims and the corresponding features of the Infringing Instrumentalities, as set forth herein. In each instance, the identified feature of the Infringing Instrumentalities performs at least substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim element.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

To the extent the chart of an asserted claim relies on evidence about certain specifically-identified Accused Instrumentalities, Push Data asserts that, on information and belief, any similarly-functioning instrumentalities also infringes the charted claim. Push Data reserves the right to amend this infringement analysis based on other products made, used, sold, imported, or offered for sale by Nordstrom. Push Data also reserves the right to amend this infringement analysis by citing other claims of the '811 patent, not listed in the claim chart, that are infringed by the Accused Instrumentalities. Push Data further reserves the right to amend this infringement analysis by adding, subtracting, or otherwise modifying content in the "Accused Instrumentalities" column of each chart.

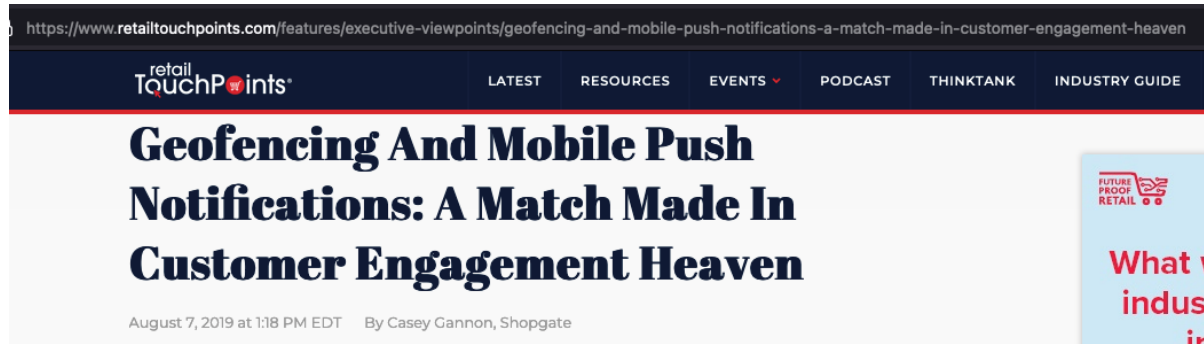
CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811



CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

Claim #1	Accused Instrumentalities
1. For use in a mobile data network environment comprising a packet switched data network, one or more network servers, a plurality of mobile units, and a plurality of wireless packet access stations coupled to the packet switched data network, wherein each wireless packet access station provides wireless access services,	<p>A method is specified for use in a mobile data network environment that uses mobile internet data services such as provided by 3G, 4G, 5G and WiFi networks.</p> <p>The network servers can be any server that provides data services such as Internet services, and mobile Internet based services.</p> <p>The mobile units can correspond to smartphones or tablets operated by users.</p> <p>The wireless packet access stations can be data nodes of 3G or 4G or 5G and/or WiFi access points.</p>
wherein a particular one of the mobile units comprises a processor, a memory, a user interface, and at least one wireless air interface comprising a wireless transmitter, a wireless receiver, and a protocol stack adapted to process wireless packet data transactions using a wireless packet data network protocol, wherein a particular user operates the particular mobile unit, and	<p>The particular mobile unit can be any smartphone or tablet, and as such, will include a processor, a memory, and a graphical user interface. Additionally it will include a WiFi air interface subsystem that will include a wireless transmitter, a wireless receiver, and a protocol stack adapted to process packet data transactions using a wireless packet data network protocol, such as wireless internet protocols that make use of protocol stacks.</p> <p>The smartphone or tablet will also typically include a second wireless interface to support 3G or 4G or 5G data services and their related wireless data interactions.</p>
the particular mobile unit is in communication with at least a particular wireless packet access	In order for the smartphone or tablet to receive the push notification, it must be in wireless communication with, for example, a 3G network, a 4G network, a 5G network, and/or a WiFi access point.

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

station when the particular mobile unit is located in a coverage area of the particular wireless packet access station,	
a method comprising:	Note that all of the limitations above are the preamble which is describing the environment in which the method actions listed below are to be practiced.
receiving a user interest indication associated with the particular user, wherein the user interest indication identifies one or more user preferences used to identify information that comports with the user interest indication;	<p>In the Nordstrom application service, user interest indication information is received by the Nordstrom application server. This user interest indication is used to identify which type of information to send to the user via Notification Messages. For example, in some systems, the user can explicitly identify specific categories of information of interest to the user. The user interest indication can also be based upon on user past actions or searches. The user interest indication can also be received from third party information providers that keep track of user interest information for micro-marketing purposes.</p> <p>For example, the application program collects information from the user's use of the application program that Nordstrom uses as an indication of the type of information that is of interest to the user, as shown below:</p>  <p>The screenshot shows a web page from retailtouchpoints.com. The URL is https://www.retailtouchpoints.com/features/executive-viewpoints/geofencing-and-mobile-push-notifications-a-match-made-in-customer-engagement-heaven. The page has a dark blue header with the 'retail TouchPoints' logo and navigation links: LATEST, RESOURCES, EVENTS, PODCAST, THINKTANK, and INDUSTRY GUIDE. The main content area has a large title 'Geofencing And Mobile Push Notifications: A Match Made In Customer Engagement Heaven' in bold black text. Below the title, it says 'August 7, 2019 at 1:18 PM EDT By Casey Gannon, Shopgate'. On the right side, there is a vertical blue banner with the text 'What's in the air' and a small icon of a shopping cart with a signal wave.</p>

December 19, 2023

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

indication, wherein the location indication identifies at least an approximate geographical location of the particular mobile unit;	This way, the Nordstrom server receives a location indication that identifies at least an approximate geographical location of the user's smartphone or tablet.
identifying an information item that comports with the user interest indication and is associated with the location identified in the location indication; and	At times, the Nordstrom server will identify information relevant to the user and also relevant to the user's location.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

<https://www.retailtouchpoints.com/features/executive-viewpoints/geofencing-and-mobile-push-notifications-a-match-made-in-customer-engagement-heaven>

[retail TouchPoints®](#)
[LATEST](#)
[RESOURCES](#)
[EVENTS ▾](#)
[PODCAST](#)
[THINKTANK](#)
[INDUSTRY GUIDE](#)

Geofencing Done Right In The Real World

According to a survey from [Localitytics](#), 42% of smartphone users said they'd use a retailer's app more if it sent them push notifications triggered by their present location. Geofencing technology is crucial to driving this type of engagement, giving retailers the ability to send contextually relevant push notifications on smartphones to help drive consumers to stores. Retailers can use geofencing-powered communication to optimize their mobile capabilities, targeting consumers where they are and staying at the top of their radar based on current location and location history.

E-BOOK

2,400 Shoppers Told Us What They Want. Are You Listening?

DOWNLOAD

MIZUNO.COM/CLICK-ADDSOURCE

ADVERTISEMENT

Mizuno USA, a leading sports and athletic goods retailer, recently updated its mobile app to drive in-store traffic to its dealers. When a customer is near a dealer's store, Mizuno can send a geofence-powered push notification directly to a user's smartphone. Once the user clicks on it, he or she will be taken directly to Google Maps to easily locate the stores that are within a specified geofence radius. The same can be done as soon as a user is near a competitor's store, steering them away with a better promotion or offer. Retailers can even create a push notification campaign targeting users at a relevant sporting event, sending participants an incentive to visit a brick-and-mortar store while creating a positive association of the brand.

Other innovative retailers and brands have also recently utilized geofencing to their advantage:

- McDonald's recently connected geofenced billboards to its in-app advertising on Waze to achieve 6.4 million mobile impressions and prompt consumers to visit a nearby location during their drive.
- A Volvo Dealership in the New York Tri-State metro has utilized geofencing initiatives to target consumers in luxury markets and competitor dealerships, as well as consumers within their own dealership, to achieve a 140% increase in foot traffic.
- [Nordstrom](#) utilizes geofencing to identify when loyal customers are within the store to offer hyper-personalized customer service.

**What w
indust
in**

EXPLORE THE RI
WORLD OF TOM

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

causing information relating to the information item to be coupled from a network server of the one or more network servers via the packet switched data network to the particular wireless packet access station so that the particular wireless packet access station can wirelessly transmit the information relating to the information item via one or more data packets	<p>As discussed in Endnote #1, the send-push notification type function calls made from the Nordstrom application server causes a push notification message to be wirelessly transmitted to the smartphone or tablet operated by a specified user.</p> <p>In the Nordstrom application, the information is caused to be sent to the mobile unit as discussed in sections that discuss the first and third method actions above.</p>
without the need to continuously maintain an active user-interactive client-server application layer session between the network server and the particular mobile unit between a first time when the user interest indication is received to a second time when the particular mobile unit receives the one or more data packets;	<p>Nordstrom's client-server session uses TLS as described in Endnote #2. The push session into the client-side app is also implemented as a TLS session as discussed in Endnote #2.</p> <p>When sent in push messages, the information relating to the information item are not a server response message sent in response to a client request message coupled from the particular mobile unit substantially just prior to the incoming communication being transmitted. For example, a push notification message is not a server response message sent in response to a client Get message in the HTML and related markup language protocols.</p>
wherein the one or more data packets are coupled at least partially via a virtual communication session implemented at one or more layers below the application layer,	<p>In Android based systems and devices, push notification messages are sent using an operating system push notification service. All push notification messages are sent over Android/FCM/GCM Push Sessions that use Transport Layer Security connections. This TLS security for push sessions is mandated by the Android operating system.</p>

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

	<p>See Endnote#2 for a discussion of the virtual session aspects of TLS. Note that TLS is located at a layer below the application layer and above the transport layer, often called the sockets layer.</p> <p>A virtual communication session is a session that has a full handshake sequence that is used to establish connection parameters, and an abbreviated handshake sequence that is used to resume the virtual session connection from an inactive or dormant state to an active state whereby new payload data can be sent via the virtual session once again.</p> <p>TLS connections are virtual sessions because they establish parameters in an initial handshake procedure to determine session parameters such as device tokens, and then reuse these virtual session parameters later to wake up and reuse the TLS session in an abbreviated handshake procedure.</p>
<p>wherein the virtual communication session is configured to be transitioned from an initial active state to an inactive state, and later to be transitioned from the inactive state back to the active state, wherein when the virtual communication session is in the active state, data related to the active user-interactive client-server application layer session and the one or more data packets can be coupled to the mobile unit via the virtual communication session.</p>	<p>In Android/FCM/GCM wireless push notification services, push notification message is sent via a virtual communication session (TLS session) that is configured to be transitioned from an initial active state to an inactive state, and later to be transitioned from the inactive state back to the active state, and when the virtual communication session is in the active state, the push notification message can be coupled to the mobile unit via the virtual communication session (TLS session, see especially Endnote #2).</p> <p>As per Endnote #1, the wireless push session is established and used over and over again over the life of the application. As per Endnote #2, TLS is used to reactivate the wireless push session each time a new push needs to be sent from the push server to the client-side app. This reactivation and push message sending is caused by the Nordstrom application server making an API call to cause the push notification message to be sent.</p>

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

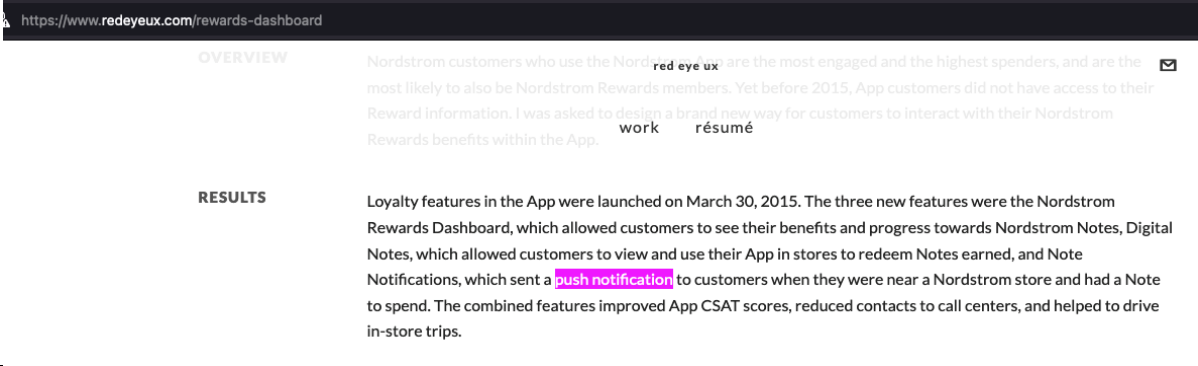
CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

Claim #5	Accused Instrumentalities
<p>5. For use in a mobile data network environment comprising a packet switched data network, one or more network servers, a plurality of mobile units including a particular mobile unit operated by a user, and a plurality of wireless packet access stations coupled to the packet switched data network, wherein each wireless packet access station provides wireless access services,</p>	<p>A method is specified for use in a mobile data network environment that uses mobile internet data services such as provided by 3G, 4G, 5G and WiFi networks.</p> <p>The network servers can be any server that provides data services such as Internet services, and mobile Internet based services.</p> <p>The mobile units can correspond to smartphones or tablets operated by users.</p> <p>The wireless packet access stations can be data nodes of 3G or 4G or 5G and/or WiFi access points.</p>
<p>wherein the particular mobile unit comprises a processor, a memory, a graphical user interface, and at least one wireless air interface comprising a wireless transmitter, a wireless receiver, and a protocol stack adapted to process wireless packet data transactions using a wireless packet data network protocol,</p>	<p>The particular mobile unit can be any smartphone or tablet, and as such, will include a processor, a memory, and a graphical user interface. Additionally it will include a WiFi air interface subsystem that will include a wireless transmitter, a wireless receiver, and a protocol stack adapted to process packet data transactions using a wireless packet data network protocol, such as wireless internet protocols that make use of protocol stacks.</p> <p>The smartphone or tablet will also typically include a second wireless interface to support 3G or 4G or 5G data services and their related wireless data interactions.</p>
<p>the particular mobile unit is configured to wirelessly receive an incoming communication from a remote application server of the one or more network servers, read an application-program identifying field contained within</p>	<p>The smartphone or tablet will run various application programs to include the Nordstrom application program after it has been downloaded.</p> <p>The Nordstrom application receives wireless push notification packets from a network server that originates at the Nordstrom server using an operating system push notification service. This corresponds to the incoming communication. For an overview of push notifications, see</p>

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

<p>the incoming communication to identify a particular application program resident on the particular mobile unit, and to present to the user via the graphical user interface a selectable indication, such that when the selectable indication is selected by a user selection, further content related to the particular application program will be downloaded to the particular mobile unit, and</p>	<p>https://www.urbanairship.com/push-notifications-explained. Endnotes #3-#6 also provide further details about the Android-specific push notification services.</p> <p>Endnote #1 provides details on how each push notification coming into the Nordstrom application includes an app-specific device token [2]. The app-specific device token will be indicative of the Nordstrom application when push notifications are sent to the Nordstrom application.</p> <p>Endnote #3 describes the main types of push notification messages. The push notification message can be displayed to the user in various ways, to include a popup message in the display of the user's smartphone or tablet, or from within a user interface supplied by the Nordstrom application.</p> <p>As discussed in Endnote #4, the push notification message may contain user-selectable information that indicates available content that can be downloaded in response to a user tap on the notification. Endnotes #5 shows one way external data may be referenced by the data object in the notification message. Endnote #6 discusses alternative ways that the user can cause external content to be downloaded, such as in response to a special user interface popped up in response to a VoIP call, or in a gaming app, or a social media app, and the like.</p> <p>The push notification message can be displayed to the user in various ways, to include a popup message in the display of the user's smartphone or tablet, or from within a user interface supplied by the Nordstrom application.</p> <p>The Nordstrom application supports push notifications. When the push notifications are enabled, push notifications are received. Push notifications cause further content related to the push notification to be downloaded or otherwise synchronized between the Nordstrom client-side App and the Nordstrom server-side application program that caused the push notification to be sent to the client-side App.</p>
---	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

	 <p>The screenshot shows a web page titled "https://www.redeyeux.com/rewards-dashboard". It has two main sections: "OVERVIEW" and "RESULTS".</p> <p>OVERVIEW: Nordstrom customers who use the Nordstrom app are the most engaged and the highest spenders, and are the most likely to also be Nordstrom Rewards members. Yet before 2015, App customers did not have access to their Reward information. I was asked to design a brand new way for customers to interact with their Nordstrom Rewards benefits within the App. Below this text are the words "work" and "résumé" with a small envelope icon.</p> <p>RESULTS: Loyalty features in the App were launched on March 30, 2015. The three new features were the Nordstrom Rewards Dashboard, which allowed customers to see their benefits and progress towards Nordstrom Notes, Digital Notes, which allowed customers to view and use their App in stores to redeem Notes earned, and Note Notifications, which sent a push notification to customers when they were near a Nordstrom store and had a Note to spend. The combined features improved App CSAT scores, reduced contacts to call centers, and helped to drive in-store trips.</p>
the particular mobile unit is in communication with at least a particular wireless packet access station when the particular mobile unit is located in a coverage area of the particular wireless packet access station,	In order for the smartphone or tablet to receive the push notification, it must be in wireless communication with, for example, a 3G network, a 4G network, a 5G network, and/or a WiFi access point.
a method comprising:	Note that all of the limitations above are the preamble which is describing the environment in which the method actions listed below are to be practiced.
causing the incoming communication to be wirelessly transmitted to the particular mobile unit, wherein the incoming communication includes the application-program identifying field that identifies the particular application program and contains information related to the further content available for	<p>The send-push notification type function calls made from the Nordstrom application server causes a push notification message to be wirelessly transmitted to the smartphone or tablet operated by a specified user.</p> <p>For details about the application-program identifying field, see the third preamble section above and Endnote #1.</p> <p>In the Nordstrom application, for example, a push notification is shown and described in the third preamble section above.</p>

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

downloading in response to the user selection,	
the incoming communication is not a server response message sent in response to a client request message coupled from the particular mobile unit substantially just prior to the incoming communication being transmitted,	<p>As discussed in the third preamble section above, the incoming communication that the Nordstrom server causes to be wirelessly transmitted is a push notification message.</p> <p>A push notification message and a push email message are not a server response message sent in response to a client request message coupled from the particular mobile unit substantially just prior to the incoming communication being transmitted. For example, a push notification message and a push email message are not a server response message sent in response to a client Get message in the HTML and related markup language protocols.</p>
the incoming communication is coupled at least partially via a virtual communication session implemented at one or more layers below the application layer,	<p>In Android based systems and devices, push notification messages are sent using an operating system push notification service. All push notification messages are sent over Android/FCM/GCM Push Sessions that use Transport Layer Security connections. This TLS security for push sessions is mandated by the Android operating system.</p> <p>See Endnote#2 for a discussion of the virtual session aspects of TLS. Note that TLS is located at a layer below the application layer and above the transport layer, often called the sockets layer.</p> <p>A virtual communication session is a session that has a full handshake sequence that is used to establish connection parameters, and an abbreviated handshake sequence that is used to resume the virtual session connection from an inactive or dormant state to an active state whereby new payload data can be sent via the virtual session once again.</p> <p>TLS connections are virtual sessions because they establish parameters in an initial handshake procedure to determine session parameters such as device tokens, and then reuse these virtual session parameters later to wake up and reuse the TLS session in an abbreviated handshake procedure.</p>

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

the virtual communication session is configured to be transitioned from an initial “ state to an inactive state, and later to be transitioned from the inactive state back to the active state, and when the virtual communication session is in the active state, the incoming communication can be coupled to the mobile unit via the virtual communication session;	<p>In Android/FCM/GCM wireless push notification services, push notification message is sent via a virtual communication session (TLS session) that is configured to be transitioned from an initial active state to an inactive state, and later to be transitioned from the inactive state back to the active state, and when the virtual communication session is in the active state, the push notification message can be coupled to the mobile unit via the virtual communication session (TLS session, see especially Endnote #2).</p> <p>As per Endnote #1, the wireless push session is established and used over and over again over the life of the application. As per Endnote #2, TLS is used to reactivate the wireless push session each time a new push needs to be sent from the push server to the client-side app. This reactivation and push message sending is caused by the Nordstrom application server making an API call to cause the push notification message to be sent.</p>
receiving a client-request packet wirelessly coupled from the particular mobile unit in response to the user selection, the client-request packet indicating a request to download the further content;	<p>In response to the user tap in the client device, the server-side application program receives the client-request packet. This client-request packet indicates a request to download the further content.</p> <p>See Endnotes #4 and #6 for a discussion of user selection of a notification which then causes downloading of the further content in response thereto.</p>
sending the further content to the mobile unit in response to the client-request packet;	<p>The Nordstrom server provides the Nordstrom application service to the Nordstrom application running on the user’s smartphone or tablet. The Nordstrom server will respond to the client-request packet by sending the further content to the Nordstrom application.</p> <p>See the third preamble section above for the specific push message and further content sent by Nordstrom server-side application.</p>
wherein the incoming communication acts as a notification to allow the user to	The incoming communication is a push message that causes user interface controls to be shown to the user that allows the user to thereby selectively download the further content.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

selectively download the further content only if the user is interested in receiving the further content.	See the third preamble section above for the user interface action that the user uses to cause the further content to be selectively downloaded into the client-side app.
---	---

Caveat: The notes and/or cited excerpts utilized herein are set forth for illustrative purposes only and are not meant to be limiting in any manner. For example, the notes and/or cited excerpts, may or may not be supplemented or substituted with different excerpt(s) of the relevant reference(s), as appropriate. Further, to the extent any error(s) and/or omission(s) exist herein, all rights are reserved to correct the same.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

Endnote#1 - app-specific device token

<https://help.pushwoosh.com/hc/en-us/articles/360000364923-What-is-a-Device-token->

What is a Device token?

Question:

What is a Device token?

Answer:

Push token (device token) - is a unique key for the app-device combination which is issued by the Apple or Google push notification gateways. It allows gateways and push notification providers to route messages and ensure the notification is delivered only to the unique app-device combination for which it is intended.

iOS device push tokens are strings with 64 hexadecimal symbols. Push token example:

03df25c845d460bcdad7802d2vf6fc1dfde97283bf75cc993eb6dca835ea2e2f

Make sure that iOS push tokens you use when targeting specific devices in your API requests are in lower case.

Android device push tokens can differ in length (usually below 255 characters), and usually start with APA... Push token example:

APA91bFoi3lMMre9G3XzR1LrF4ZT82_15MsMdEICogXSLB8-
MrdkRuRQFwNI5u8Dh0cI90ABD3BOKnxkEla8cGdisbDHl5cVlkZah5QUhSAxxz4Roa7b4xy9tvx9iNSYw-
eXBYYd8k1XKf8Q_Qq1X9-x-U-Y79vdPq

.....

Note: The Android device push tokens correspond to the app-specific device token terminology used in the claim charts.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

<https://dev.to/jakubkoci/react-native-push-notifications-313i>

In the above Architecture, the “backend” corresponds to the backend of the Application Server. The Device token corresponds to a specific App running on a specific device. That is what is meant by the app-specific device token in the claim charts.

<https://firebase.google.com/docs/cloud-messaging/android/first-message>

“Access the registration token

To send a message to a specific device, you need to know that device's registration token. Because you'll need to enter the token in a field in the Notifications console to complete this tutorial, make sure to copy the token or securely store it after you retrieve it.

On initial startup of your app, the FCM SDK generates a registration token for the client app instance. If you want to target single devices or create device groups, you'll need to access this token by extending `FirebaseMessagingService` and overriding `onNewToken`.

This section describes how to retrieve the token and how to monitor changes to the token. Because the token could be rotated after initial startup, you are strongly recommended to retrieve the latest updated registration token.

The registration token may change when:

- The app deletes Instance ID
- The app is restored on a new device
- The user uninstalls/reinstall the app
- The user clears app data.”

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

<https://firebase.google.com/docs/cloud-messaging/concept-options>

For example, here is a JSON-formatted notification message in an IM app. The user can expect to see a message with the title "Portugal vs. Denmark" and the text "great match!" on the device:

```
{
  "message": {
    "token": "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...", □- App-specific token
    "notification": {
      "title": "Portugal vs. Denmark",
      "body": "great match!"
    }
  }
}
```

<https://firebase.google.com/docs/cloud-messaging/android/client>

Retrieve the current registration token

When you need to retrieve the current token, call `FirebaseInstanceId.getInstance().getInstanceId()`:

```
FirebaseInstanceId.getInstance().getInstanceId()
    .addOnCompleteListener(new OnCompleteListener<InstanceIdResult>() {
        @Override
        public void onComplete(@NonNull Task<InstanceIdResult> task) {
```

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

```
if (!task.isSuccessful()) {  
    Log.w(TAG, "getInstanceId failed", task.getException());  
    return;  
}  
  
// Get new Instance ID token  
String token = task.getResult().getToken();  
  
// Log and toast  
String msg = getString(R.string.msg_token_fmt, token);  
Log.d(TAG, msg);  
Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();  
}  
});
```

MainActivity.java

Monitor token generation

The onNewToken callback fires whenever a new token is generated.

/**

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

* Called if InstanceID token is updated. This may occur if the security of
 * the previous token had been compromised. Note that this is called when the InstanceID token
 * is initially generated so this is where you would retrieve the token.
 */

@Override

```
public void onNewToken(String token) {
    Log.d(TAG, "Refreshed token: " + token);

    // If you want to send messages to this application instance or
    // manage this apps subscriptions on the server side, send the
    // Instance ID token to your app server.

    sendRegistrationToServer(token);
}
```

After you've obtained the token, you can send it to your app server and store it using your preferred method. See the Instance ID API reference for full detail on the API.

Endnote#2 - Transport Layer Security and Virtual Sessions

<https://developer.android.com/training/articles/security-ssl>

“The Secure Sockets Layer (SSL)—now technically known as Transport Layer Security (TLS)—is a common building block for encrypted communications between clients and servers.”

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

<https://android-developers.googleblog.com/2018/04/protecting-users-with-tls-by-default-in.html>

“Android is committed to keeping users, their devices, and their data safe. One of the ways that we keep data safe is by protecting all data that enters or leaves an Android device with Transport Layer Security (TLS) in transit.

http://abbas.rpanah.ir/publications/conext2017_tls_paper.pdf - However, other protocols such as secure email (42 apps) and Google’s Cloud Messaging service for push notifications (9 apps) [11, 47] also use TLS. A History of TLS Support in Android: Android has supported TLS 1.0 since its first version released in 2008 and TLS 1.1 and TLS 1.2 since 2012.

<https://developer.ibm.com/customer-engagement/docs/watson-marketing/ibm-engage-2/tls-1-2-migration-for-mobile-push-clients/>

What will happen on devices that are unable to support TLS 1.2?

Devices which do not support TLS 1.2 will be unable to connect to our WCA servers. This will prevent users of those devices from:

- Registering new mobile user IDs
- Updating push tokens
- Receiving inbox messages
- Receiving In-app messages

As the above link shows, the creation of the App IDs of Endnote #1 are linked to the TLS protocol being run on the TLS-enabled Push-Notification channel.

<https://tools.ietf.org/html/rfc5246> - TLS 1.2 - Has fast session resumption, section F.1.4

<https://tools.ietf.org/html/rfc5077> - This is the version introduces server tickets, like Android Device Push Tokens

Abstract

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

This document describes a mechanism that enables the Transport Layer Security (TLS) server to resume sessions and avoid keeping per-client session state. The TLS server encapsulates the session state into a ticket and forwards it to the client. The client can subsequently resume a session using the obtained ticket.

3. Protocol

This specification describes a mechanism to distribute encrypted session-state information in the form of a ticket. The ticket is created by a TLS server and sent to a TLS client. The TLS client presents the ticket to the TLS server to resume a session.

Endnote#3 – Android/FCM/GCM Notification Message Types

<https://firebase.google.com/docs/cloud-messaging/concept-options>

About FCM messages

Firebase Cloud Messaging (FCM) offers a broad range of messaging options and capabilities. The information in this page is intended to help you understand the different types of FCM messages and what you can do with them.

Message types

With FCM, you can send two types of messages to clients:

- Notification messages, sometimes thought of as "display messages." These are handled by the FCM SDK automatically.
- Data messages, which are handled by the client app.

Notification messages contain a predefined set of user-visible keys. Data messages, by contrast, contain only your user-defined custom key-value pairs. Notification messages can contain an optional data payload. Maximum payload for both message types is 4KB, except when sending messages from the Firebase console, which enforces a 1024 character limit.

Use scenario How to send

Notification message FCM automatically displays the message to end-user devices on behalf of the client app. Notification messages have a predefined set of user-visible keys and an optional data payload of custom key-value pairs. 1. In a trusted environment such as Cloud Functions or your app server, use the Admin SDK or the FCM Server Protocols: Set the notification key. May have optional data payload. Always collapsible.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

2. Use the Notifications composer: Enter the Message Text, Title, etc., and send. Add optional data payload by providing Custom data.

Data message Client app is responsible for processing data messages. Data messages have only custom key-value pairs with no reserved key names (see below). In a trusted environment such as Cloud Functions or your app server, use the Admin SDK or the FCM Server Protocols: Set the data key only.

Use notification messages when you want FCM to handle displaying a notification on your client app's behalf. Use data messages when you want to process the messages on your client app.

FCM can send a notification message including an optional data payload. In such cases, FCM handles displaying the notification payload, and the client app handles the data payload. (...)

Notification messages are delivered to the notification tray when the app is in the background. For apps in the foreground, messages are handled by a callback function. (...)

App behavior when receiving messages that include both notification and data payloads depends on whether the app is in the background or the foreground (...).

- When in the background, apps receive the notification payload in the notification tray, and only handle the data payload when the user taps on the notification. [Data messages go straight to the client app and wake it up. That is, both the data message and its optional data payload are delivered directly to the client app with no need for the user to tap on the user interface, although sounds may be triggered and badges and short messages may be flash-displayed to alert the user.]
- When in the foreground, your app receives a message object with both payloads available

<https://firebase.google.com/docs/cloud-messaging/android/receive>

Handling messages

(...) [onMessageReceived() = executed when a notification message or a data message is received.]

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

- Notification messages delivered when your app is in the background. In this case, the notification is delivered to the device's system tray. A user tap on a notification opens the app launcher by default.
- Messages with both notification and data payload, when received in the background. In this case, the notification is delivered to the device's system tray, and the data payload is delivered in the extras of the intent of your launcher Activity. In summary:

App state	Notification	Data	Both
Foreground	onMessageReceived	onMessageReceived	onMessageReceived
Background	System tray	onMessageReceived	Notification: system tray

Data: in extras of the intent.

Endnote#4 – Actions Taken Upon Notification-Message User Tap

<https://developer.android.com/guide/topics/ui/notifiers/notifications>

Notifications Overview

A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification. (...)

Notification actions

Although it's not required, every notification should open an appropriate app activity when tapped. In addition to this default notification action, you can add action buttons that complete an app-related task from the notification (often without opening an activity), as shown in figure 9.

Figure 9. A notification with action buttons

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

Starting in Android 10 (API level 29), the platform can automatically generate action buttons with suggested intent-based actions.

[Note: When the user taps the notification or an action button on the notification, an Intent can be executed as an activity, a service, or a receive-broadcast operation. The Intent, delivered as a Pending Intent identifies the program code to be executed upon user tap of the Notification. The Intent and Pending Intent executed upon user-tap the Notification is described below.]

<https://developer.android.com/guide/components/intents-filters>

Intents and Intent Filters

An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use cases:

- Starting an activity

An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to `startActivity()`. The Intent describes the activity to start and carries any necessary data. (...)

- Starting a service

A Service is a component that performs operations in the background without a user interface. (...)

(...) You can start a service to perform a one-time operation (such as downloading a file) by passing an Intent to `startService()`. The Intent describes the service to start and carries any necessary data.

- Delivering a broadcast

A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. (...)

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

Using a pending intent

A `PendingIntent` object is a wrapper around an `Intent` object. The primary purpose of a `PendingIntent` is to grant permission to a foreign application to use the contained `Intent` as if it were executed from your app's own process.

Major use cases for a pending intent include the following:

- Declaring an intent to be executed when the user performs an action with your Notification (the Android system's `NotificationManager` executes the `Intent`). (...)

[Y]ou must declare the intended component type when you create the `PendingIntent` by calling the respective creator method:

- `PendingIntent.getActivity()` for an `Intent` that starts an `Activity`.
- `PendingIntent.getService()` for an `Intent` that starts a `Service`.
- `PendingIntent.getBroadcast()` for an `Intent` that starts a `BroadcastReceiver`.

For more information about using pending intents, see the documentation for each of the respective use cases, such as in the Notifications and App Widgets API guides.

[The Notification Message can carry a `PendingIntent` object, such that upon user tap, an activity or service is launched. The activity generally corresponds to a UI screen in the client app, while the service corresponds to a background process in the app, as used to download a file or the like.]

<https://developer.android.com/training/notify-user/build-notification>

Set the notification's tap action

Every notification should respond to a tap, usually to open an activity in your app that corresponds to the notification. To do so, you must specify a content intent defined with a `PendingIntent` object and pass it to `setContentIntent()`. (...)

To add an action button, pass a `PendingIntent` to the `addAction()` method. This is just like setting up the notification's default tap action, except instead of launching an activity, you can do a variety of other things such as start a `BroadcastReceiver` that performs a job in the background so the action does not interrupt the app that's already open.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

Endnote#5 – URI Data Field in Notification and Data Messages

<https://developer.android.com/guide/components/intents-filters>

The primary information contained in an Intent is the following:

Component name - The name of the component to start. (...)

Action - A string that specifies the generic action to perform (such as view or pick). (...)

Data -The URI (a Uri object) that references the data to be acted on and/or the MIME type of that data. The type of data supplied is generally dictated by the intent's action. For example, if the action is ACTION_EDIT, the data should contain the URI of the document to edit. (...)

Category - A string containing additional information about the kind of component that should handle the intent. (...)

Extras - Key-value pairs that carry additional information required to accomplish the requested action. Just as some actions use particular kinds of data URIs, some actions also use particular extras. (...)

[Note, the Intent object, passed in the PendingIntent object in the Notification Message includes a Data field which is a Uri object. (URI = Uniform Resource Identifier.) The Uri object unambiguously identifies items such as local resources in the client, external files stored on the application server, or external data sources in an external content provider database. The Uri object can resolve to a MIME type, and can also be resolved to find external information like web pages from content providers and the like. The links below explain how build the Uri object and how the Uri object can resolved.]

<https://developer.android.com/training/app-links/deep-linking>

Create Deep Links to App Content

When a clicked link or programmatic request invokes a web URI intent, the Android system tries each of the following actions, in sequential order, until the request succeeds:

1. Open the user's preferred app that can handle the URI, if one is designated.

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

2. Open the only available app that can handle the URI.
3. Allow the user to select an app from a dialog.

Follow the steps below to create and test links to your content. You can also use the App Links Assistant in Android Studio to add Android App Links.

To create a link to your app content, add an intent filter that contains these elements and attribute values in your manifest:

`<action>` (...)

`<data>` Add one or more `<data>` tags, each of which represents a URI format that resolves to the activity. At minimum, the `<data>` tag must include the `android:scheme` attribute.

You can add more attributes to further refine the type of URI that the activity accepts. For example, you might have multiple activities that accept similar URIs, but which differ simply based on the path name. In this case, use the `android:path` attribute or its `pathPattern` or `pathPrefix` variants to differentiate which activity the system should open for different URI paths. (...)

`<intent-filter>`

...

`<data android:scheme="https" android:host="www.example.com" />`

`<data android:scheme="app" android:host="open.my.app" />`

`</intent-filter>`

It might seem as though this supports only `https://www.example.com` and `app://open.my.app`. However, it actually supports those two, plus these: `app://www.example.com` and `https://open.my.app`.

Once you've added intent filters with URIs for activity content to your app manifest, Android is able to route any Intent that has matching URIs to your app at runtime.

<https://developer.android.com/reference/android/content/ContentResolver>

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

[The Uri object can be evaluated to see what the Uri is referencing. Some example classes and public methods are given below.]

Nested classes

```
class acquireContentProviderClient(Uri uri)
```

Returns a ContentProviderClient that is associated with the ContentProvider that services the content at uri, starting the provider if necessary.

```
call(Uri uri, String method, String arg, Bundle extras)
```

Call a provider-defined method.

```
requestSync(Account account, String authority, Bundle extras)
```

Start an asynchronous sync operation

<https://firebase.google.com/docs/cloud-messaging/concept-options>

Here is an example of a normal priority message sent via the FCM HTTP v1 protocol to notify a magazine subscriber that new content is available to download:

```
{
  "message": {
    "topic": "subscriber-updates",
    "notification": {
      "body": "This week's edition is now available.",
      "title": "NewsMagazine.com",
    },
  },
}
```


CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

```
"data" : {  
  "volume" : "3.21.15",  
  "contents" : "http://www.news-magazine.com/world-week/21659772"  
},  
"android": {  
  "priority": "normal"  
},  
"apns": {  
  "headers": {  
    "apns-priority": "5"  
  }  
},  
"webpush": {  
  "headers": {  
    "Urgency": "high"  
  }  
}  
}
```

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

Endnote#6 – Background Refresh and other Activities in response to Data Messages

<https://support.kochava.com/sdk-integration/android-sdk-integration/android-push-notification/>

onMessageReceived:

Kochava Push uses the FCM Data Message format. This format ensures all push messages are received by the “onMessageReceived” method no matter if the app is in the foreground or background. The fields included in the RemoteMessage data are as follows. A Kochava push will always include the “kochava” key, other fields may vary.

1. kochava – Campaign tracking information to be added to the notification bundle and sent with the Push Open event.
2. silent – If this key is present the push was used for tracking uninstalls and should be ignored.
3. title – Notification title.
4. message – Notification message body.
5. icon_resource_id – Mapping string to an app internal drawable resource.
6. link – Launch deeplink Uri.

[The link above explains that a FCM Data Message will cause onMessageReceived() to fire/execute in an app when the app is in the background mode. See Endnote #1 for a discussion of the FCM Data Message. That is, when an app is in background mode, a FCM Notification Message will go to the notifications tray, but the FCM Data Message will cause the app to awaken by activating the

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

onMessageReceived() method. Depending on how onMessageReceived() is configured, this can cause background application refresh to occur, like the client syncing a mailbox or a chat session with a server, or could cause an application like a VoIP telecommunications app to be launched.]

<https://www.semicolonworld.com/question/44020/how-to-handle-notification-when-app-in-background-in-firebase>

How to handle notification when app in background in Firebase

When the app is in background and notification arrives then the default notification comes and doesn't run my code of onMessageReceived.

Here is my onMessageReceived code. This invokes if my app is running on foreground, not when app in background. How to run this code when the app is in background too?

- June 2018 Answer -

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
 U.S. Patent No. 7,212,811

You have to make sure there is not a "notification" keyword anywhere in the message. Only include "data", and the app will be able to handle the message in onMessageReceived, even if in background or killed.

Using Cloud Functions:

```
const message = {
  token: token_id, // obtain device token id by querying data in firebase
  data: {
    title: "my_custom_title",
    body: "my_custom_body_message"
  }
}
```

```
return admin.messaging().send(message).then(response => {
  // handle response
});
```

Then in your onMessageReceived(), in your class extending com.google.firebase.messaging.FirebaseMessagingService :

```
if (data != null) {
  Log.d(TAG, "data title is: " + data.get("title"));
  Log.d(TAG, "data body is: " + data.get("body"));
}
```

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

// build notification using the body, title, and whatever else you want.

[The link below answers the same question as the above link. The bottom line is to use a data message and to use OnMessageReceived() to get the content out of the data message and to take actions based on the extracted data.]

<https://stackoverflow.com/questions/37711082/how-to-handle-notification-when-app-in-background-in-firebase>

[The links below explain how data messages can be prioritized to wake up an app and schedule a worker process to perform background sync with a server side application process like an email box or a chat session or a gaming session, a messaging app, or a maps type app.

<https://firebase.google.com/docs/cloud-messaging/concept-options>

You have two options for assigning delivery priority to downstream messages on Android: normal and high priority. Delivery of normal and high priority messages works like this:

- Normal priority. This is the default priority for data messages. Normal priority messages are delivered immediately when the app is in the foreground. When the device is in Doze, delivery may be delayed to conserve battery. For less time-sensitive messages, such as notifications of new email, keeping your UI in sync, or syncing app data in the background, choose normal delivery priority.

When receiving a normal priority message on Android that requests a background data sync for your app, you can schedule a task with WorkManager to handle it when the network is available.

- High priority. FCM attempts to deliver high priority messages immediately, allowing the FCM service to wake a sleeping device when necessary and to run some limited processing (including very limited network access). (...)

If you need to sync for additional in-app content on Android, you can schedule a task with WorkManager to handle that in the background.

<https://developer.android.com/topic/libraries/architecture/workmanager>

Use WorkManager for Deferrable and Reliable Work

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

WorkManager is intended for work that is deferrable—that is, not required to run immediately—and required to run reliably even if the app exits or the device restarts. For example:

- Sending logs or analytics to backend services
- Periodically syncing application data with a server

[See also:] <https://developer.android.com/topic/libraries/architecture/workmanager/basics>

[High priority push messages can also be used, for example to alert the user to an incoming VoIP call.]

<https://documentation.onesignal.com/docs/voip-notifications>

Android VoIP Setup

Android does not have the concept of "VoIP Push" the same as iOS. Notifications will just work, including data-only messages where you can start an Activity instead of showing a push.

If you need to call custom class for showing native UI for example, you would use the Android NotificationExtenderService so you can override the notification and show your custom UI or start your custom Activity.

[The activity started in response to the data message could be selected from the android call processing API as linked below.]

<https://developer.android.com/guide/topics/connectivity/telecom/selfManaged>

[Android background mode processing is described in the links below]

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF NORDSTROM
U.S. Patent No. 7,212,811

<https://ting.com/blog/ting-tip-for-android-control-which-apps-use-background-mobile-data/>

<https://swiftsenpai.com/testing/send-silent-push-notifications/>

<https://developer.android.com/training/notify-user/navigation?>

[Android CompactNotifications can be used to start any desired activity from a compact mode data message.]

<https://documentation.onesignal.com/docs/service-extensions#notification-extender-service>

Notification Extender Service

Android

Note! This requires writing native Android code

Set up the NotificationExtenderService if you want to do one of the following:

- Receive data in the background with or without displaying a notification.
- Override specific notification settings depending on client side app logic such as custom accent color, vibration pattern, or other any other NotificationCompat options available. See Android's documentation on the NotificationCompat options.



OneSignal also supports sending additional data along with a notification as key value pairs. You can read this additional data when a notification

<https://developer.android.com/reference/androidx/core/app/NotificationCompat>